KIRCHHOFF-
INSTITUTE
FOR PHYSICS

STiC2

User Guide

Version 3

Authors
Alejandro Gil & Yonathan Munwes

March 4, 2014

Contents

Chapter 1.

What's in the package

1.1. Package description

In the package you received the following components:

1. STiC test board
2. main board
3. FPGA board
4. USB/mini-USB cable, for connecting the FPGA to the PC (the FPGA is powered by the USB)
5. USB a-male to a-female extension cable (to configure the Avnet Spartan-6 LX9 FPGA, can also power the FPGA). The user can choose between this connection to No.4.
6. power cable (for low voltage)
7. High Voltage (HV) cable
8. software package

Figure 1.1 shows the different package components.

1.2. System requirements

In order to control the FPGA and the STiC2 chip with your personal computer, a dedicated Graphical User Interface (GUI) was developed. The system requirements are:

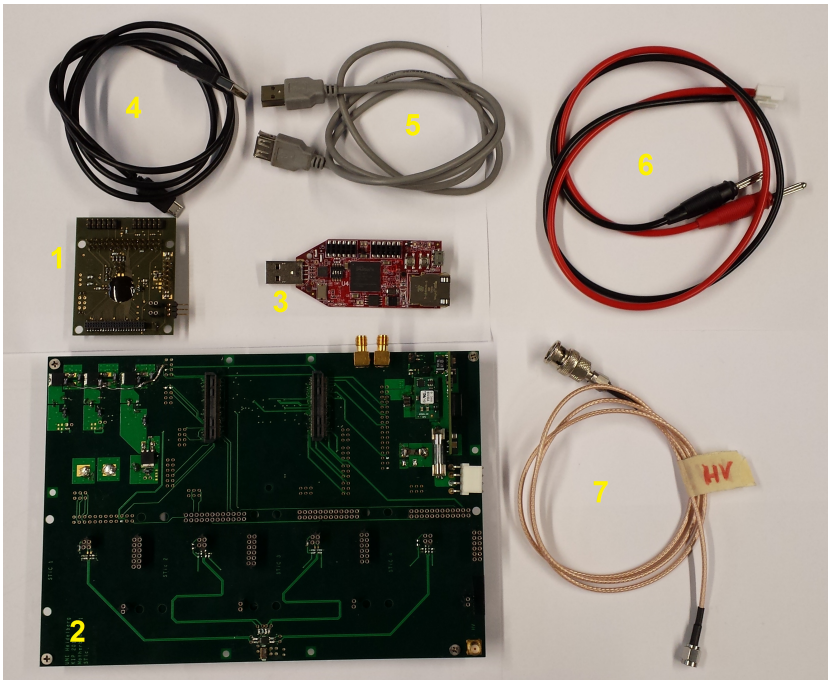


Figure 1.1.: The different package components. The numeration corresponds to the list in the text.

- linux machine with Debian 6.0 or newer version (recommended version 7.0, gtk+3.0 already embedded)
- ISE design suite - Xilinx, for using SDK kit 14.5 build $SDK\_P.58f$
  (download from `http://www.xilinx.com/products/design-tools/ise-design-suite/`)
  If you are using older version you may need to install digilent adept/plugin for xilinx.
  The package can be found in digilent website:
  Digilent Adept runtime and utility:
  `http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,828&Prod=ADEPT2`
  Digilent Plugin for Xilinx Tools:
  `http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,768&Prod=DIGILENT-PLUGIN`
- ROOT version 5.34/00, used for building the output data, online monitoring and analysis can be download from `http://root.cern.ch/drupal/`

These are the recommended requirements, if using different versions, the software package will need to recompile. In addition also the user will need a HUB switch for connecting the FPGA and the PC to the same network. The recommended HUB is DES-1008D from D-Link. **Before starting please check the all the above system requirements are fulfilled!**

Chapter 2.

System overview

If you already familiarize with the system you can go directly to Section 4

The DAQ system demonstrator for the STiC is based on three printed circuits boards, the STiC test board, the main board and the FPGA board.

2.1. STiC2 test board

The STiC chip is bonded directly to a 5 $cm \times 5$ cm 6-layers test PCB which provides adequate interface connections to the outside world. Figure 2.1 shows the PCB connections.

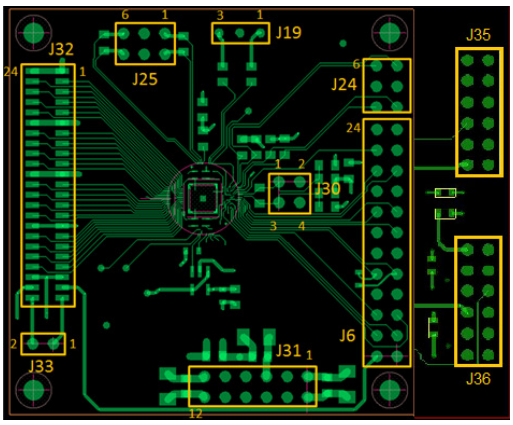


Figure 2.1.: 6-layers test PCB used for STiC.

Table 2.1 lists the relevant connection to the test board

Table 2.1.: STiC test board PCB connections.

Function	Reference	Description
Input connector	J32	Connector to plug the analog inputs signals from the 16 detectors. Each detector can be connected either in single-ended or differential mode.
Interface connector	J6	Connector for digital signals to the FPGA board that provides the data readout and the SPI configuration.
Power connector	J31	Connector for the power voltages: 3.3 V digital, 1.8 V analog and 1.8 V digital
TDC test input	J25	Input of the TDC (only for testing purposes).
Analog channel input	J19	Input to the 16th channel, which is only analog (without TDC).
Analog channel output	J24	Connector to readout the signals (energy and time) directly from the analog LVDS differential output (channel number 16).
TDC clk input	J30	622Mhz clock, which is used as a coarse-counter for each TDC.
Direct connection to FPGA	J35	Connection to FPGA J4 (see Figure 2.5).
Direct connection to FPGA	J36	Connection to FPGA J5 (see Figure 2.5).

The connector used on the board for the input for the 16 channels (J32) is SAMTEC FLE-123-01-G-DV. A possible mate for this connector is the SAMTEC FTS-123-01-F-DV. The detail input connection is listed in A (Table B.1)

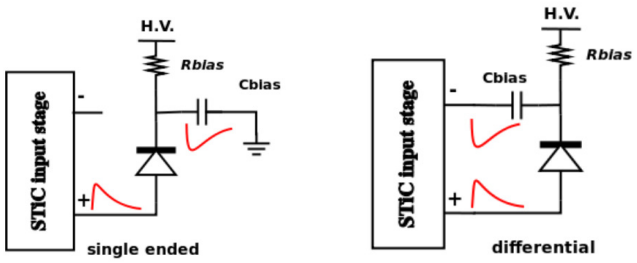
STiC is designed to work with positive signal[1] (current flowing into the chip) connected to the positive input terminal. STiC allows two possible readout modes: single ended and differential. In Figure 2.2 the two diagram for the two modes are presented, where the typical values for discrete Hamamatsu MPPCs are $R_{bias} = 10\ K$ and $C_{bias} = 100\ nF$.

The arrangement of the power connector (J31) is listed in A (Table A.2).

The interface connector (J6) is a 24-pin double row generic 2.55 mm pitch header connector, used for readout of the digital data, slow control, test signal outputs, and analog channel outputs. See its detail connection in A (Table A.3).

The connector J25 is used only for testing the TDC. The way to test is to inject input signal and check if the output signal has the same value for all pulses, this test checks if the PLL is

[1]Negative input is also possible but the energy data output will lose its linear relation with the input energy.



(a) (b)

Figure 2.2.: STiC operational modes, in (a) single ended mode (b) differential mode.

locked. If it is not locked the signal will have a distribution around the correct value. The J25 connectors can be seen in A (Table A.4).

For the analog channel we use input connector J19 and output connector J24. **Please note: J24 has 2 differential outputs**
This part is used only for testing, and it has only one channel (see A (Table A.5, A.6)). In Figure 2.3, an example of the use of the analog chaanel is presented. The red signal shows the analog signal applied to the input of the analog channel using charge injection. The blue signal is the trigger signal from the pulse generator (synchronized with the input signal). The yellow signal is the timing signal generated by STiC. This way one can check that the STiC output is generating the expected output.

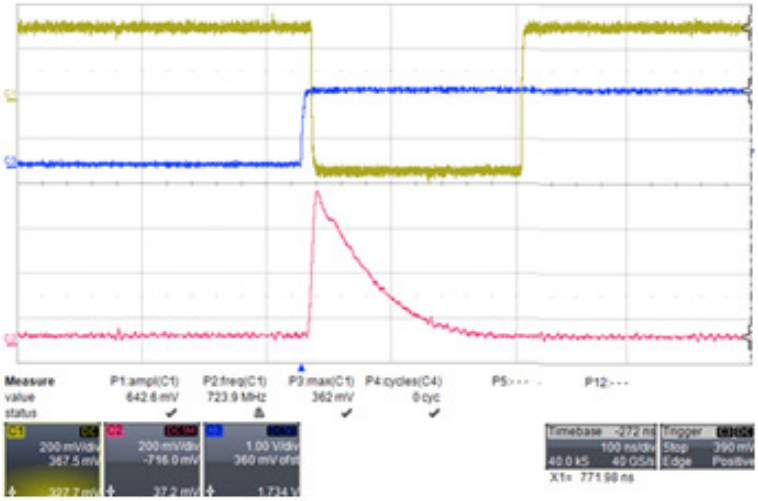


Figure 2.3.: Scope screen shot of the analog channel used for testing STiC. In red - analog signal applied in to the input analog channel, blue - trigger signal from pulse generator, and in yellow - the signal generated by STiC.

The last connector is the TDC CLK input connector (J30). It is input reference clock for the PLL. The detail connection is listed in A (Table A.7).

2.2. Main board

The main board allows to connect a maximum of 4-STiC boards, which are synchronized through the main board. The layout of the board is shown in Figure 2.4.

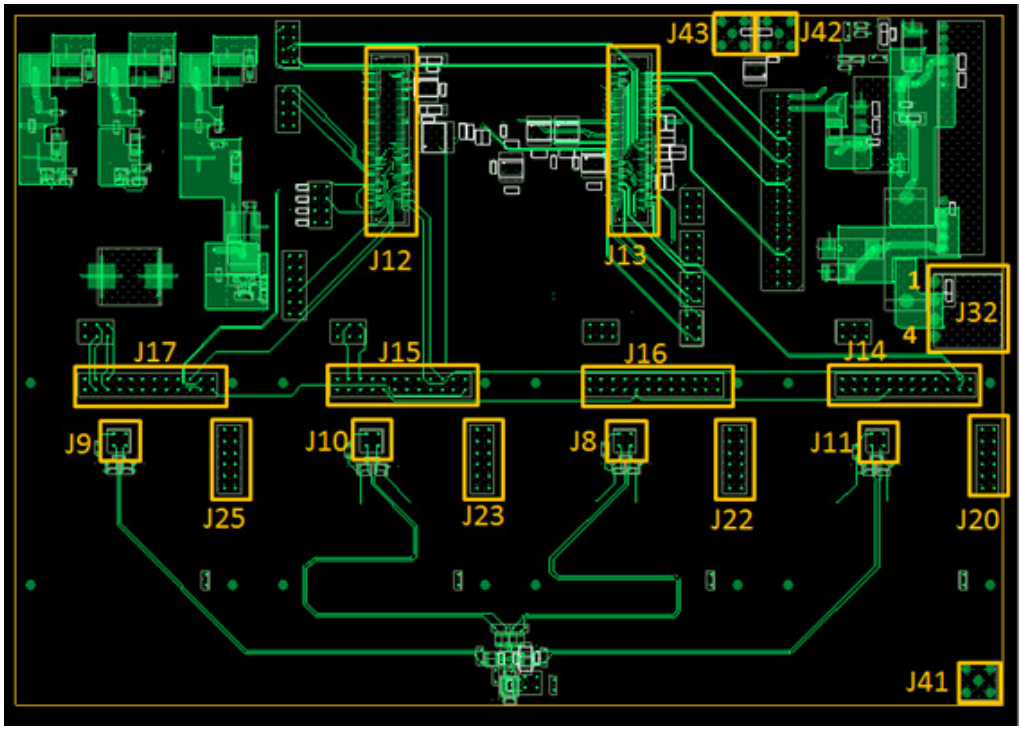


Figure 2.4.: Main board layout.

In this Figure, the different connectors to STiC and the location of the different blocks on the board are presented. The functionality of the different connectors are the following:

- **J14-J17** - digital interface connectors to STiC. Have the same pin layout as J6 (see A.3).
- **J20,J22,J23,J25** - power connectors to STiC. Have the same pin layout as J31 (see A.2).
- **J8,J9,J10,J11** - TDC clk input connectors (622 MHz clock).
- **J42,J43** - connectors used for testing purposes.
- **J12,J13** - connectors to the FPGA board. The pin layout is not required for the user.

- **J41** - input bias voltage for the SiPMs connected to STiC (typically $\approx 70\ V$).
- **J32** - power connector (12 V_{DC})

2.3. FPGA

The FPGA in the package is the Avnet Spartan-6 FPGA LX9 MicroBoard. The pin layout of the FPGA is shown in Figure 2.5.

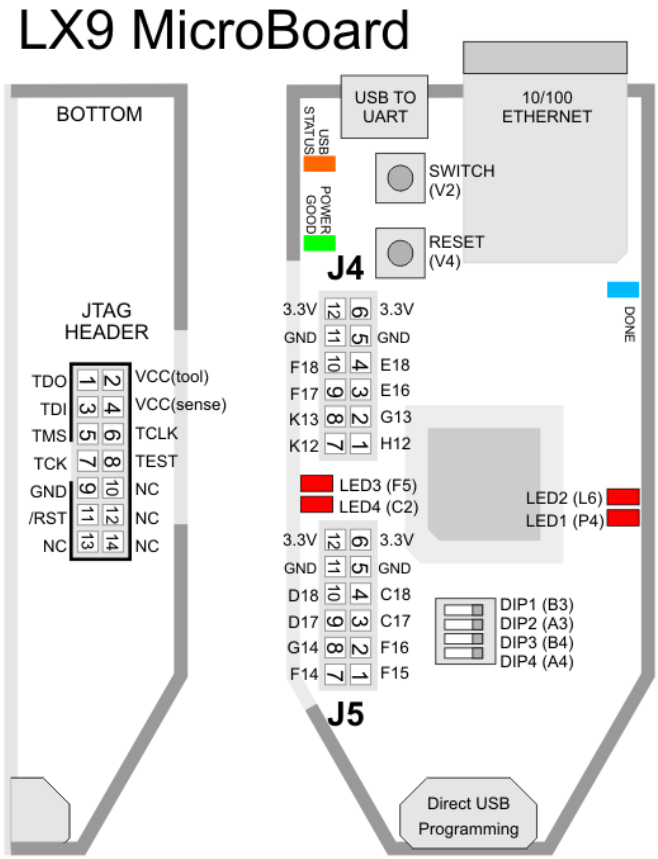


Figure 2.5.: Spartan-6 FPGA LX9 MicroBoard pin layout.

The FPGA will be connected directly to the STiC test board, and will be used for the configuration of STiC and also for readout of STiC output.

A detail description of the FPGA can be found in http://opencores.org/websvn,filedetails?repname=openmsp430&path=%2Fopenmsp430%2Ftrunk%2Ffpga%2Fxilinx_avnet_lx9microbard%2Fdoc%2FXilinx_Spartan-6_LX9_MicroBoard_Rev_B2_Hardware_User_Guide.pdf

Chapter 3.

Software

There are two software in this package, one is the Graphical User Interface (GUI) and the second is the Data Acquisition (DAQ). The GUI is used in order to change the different parameters of the STiC, for example changing the thresholds for each channel. The DAQ software is used for running the STiC to collect data, running the online monitoring and saving the data into root files.

3.1. GUI

The STiC can be connected to the PC in two ways:

- using an Ethernet cable interface through the micro SPARTAN6 board and a HUB-switch.
- directly using USB through the main board (not available at the moment).

Please make sure you use the appropriate version of files according to your interconnection method.

The two Figures 3.1, 3.2 show the window of the GUI with some default DAC values, one for "general & TDC", and the second for "Channel" (can be chosen from the bottom frame of the GUI, marked red on the Figures)

NOTE: After switching on, a reset must be applied to the chip for proper operation (via button "Chip Data Reset").

The first window is used for setting the TDC DACs, which is common for all channels. Table 3.1 summarize all the window functions.

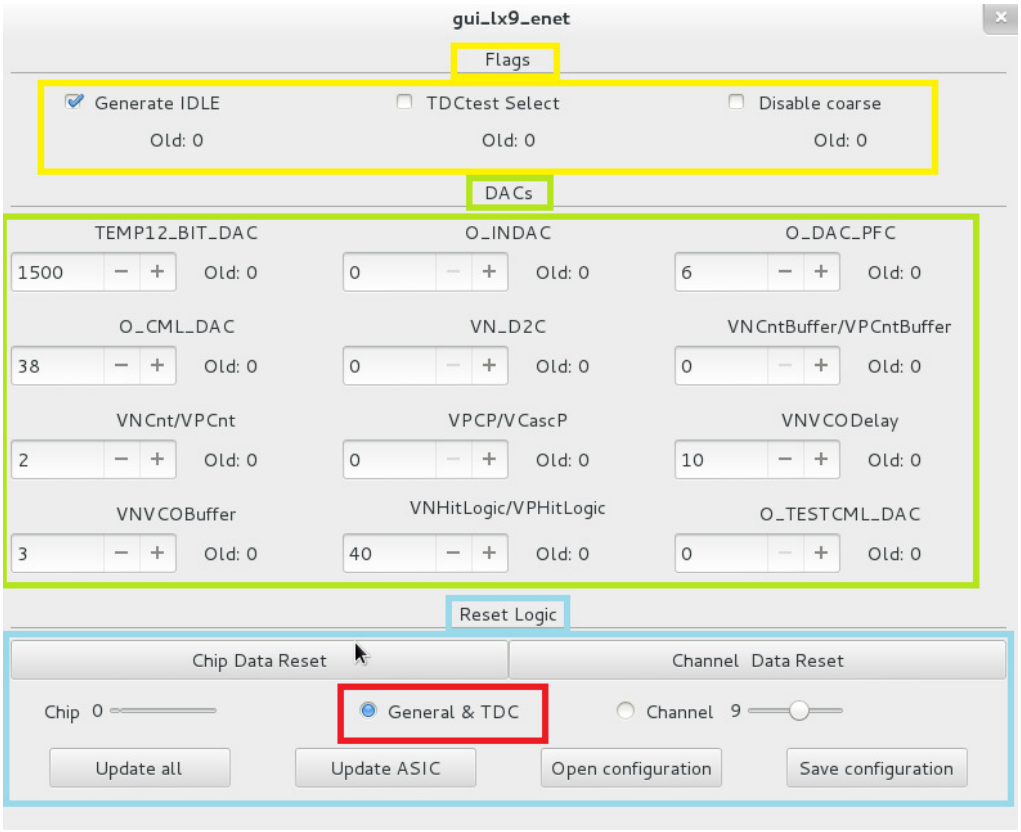


Figure 3.1.: GUI screenshot for "general & TDC" option.

It is important to select "General & TDC" settings and press the button "Chip Data Reset" and/or "Channel Data Reset" at least one time after switching on the chip. Sometimes these buttons can help to recover the normal functionality of the chip when it does not seem to behave correctly (i.e: after a power supply cut or setting very wrong DAC values).

After setting all the different DACs it is recommended to save your settings using the "Save configuration" button, for later use.

The second window is used for setting a specific channel for different thresholds, masking etc. The full functionality of this GUI is summarized in Table 3.2.

Every time any DAC parameter is changed to a different value, the parameters must be loaded into the chip by pressing "Update all" (updates all chips) or "Update ASIC", to configure only the selected chip.

IMPORTANT: The GUI does not show any error messages. However, if the configuration does not reach STiC. however, one can check that the GUI is working by checking the "Old" value shown beside every parameter box. If the "Old" parameter changes to the new one

Table 3.1.: GUI list function for selection of "general & TDC".

Frame	Name	Value range	Function	Default value
	Generate IDLE	Select	Generates sync signal continuously between data frames	
Flags	TDCtest Select	Select	Enables the direct TDC input on channel 15	
(marked yellow 3.1)	Disable coarse	Select	Disable the coarse counter	
	TEMP12_BIT_DAC	0-4095	Bias for the latches in the TDC (400 mV internal)	1500
	O_INDAC	0-63	Internal bias DAC	0
	O_DAC_PFC	0-63	Internal bias DAC	0
	O_CML_DAC	0-63	Internal bias DAC	0
	VN_D2C	0-63	Internal bias DAC	0
DACs	VNCntBuffer/VPCntBuffer	0-63	Internal bias DAC	0
(marked green 3.1)	VNCnt/VPCnt	0-63	Internal bias DAC	0
	VPCP/VCascP	0-63	Internal bias DAC	0
	VNVCODelay	0-63	Internal bias DAC	0
	VNVCOBuffer	0-63	Internal bias DAC	0
	VNHitLogic/VPHitLogic	0-63	Internal bias DAC	0
	O_TESTCML_DAC	0-63	Internal bias DAC	0
	Chip Data Reset			
	Channel Data Reset			
	Chip	0-3	Select the chip you want to configure	
Reset Logic	Update all		Update all ASICs with the selected DACs configuration	
(marked blue 3.1)	Update ASIC		Update the selected chip only	
	Open configuration		Upload configuration file	
	Save configuration		Save current configuration to a file	

after pressing the "Update" button, then the GUI is working and STiC is getting the values correctly. If the "Old" value does not change, or is set to any other value different to the one set, then STiC is not getting the parameters. In this case check the connection and make sure that STiC is powered on, the FPGA board programmed and the connection from STiC to the FPGA is correct. Also check that the cable (USB or Ethernet) is connected to the computer.

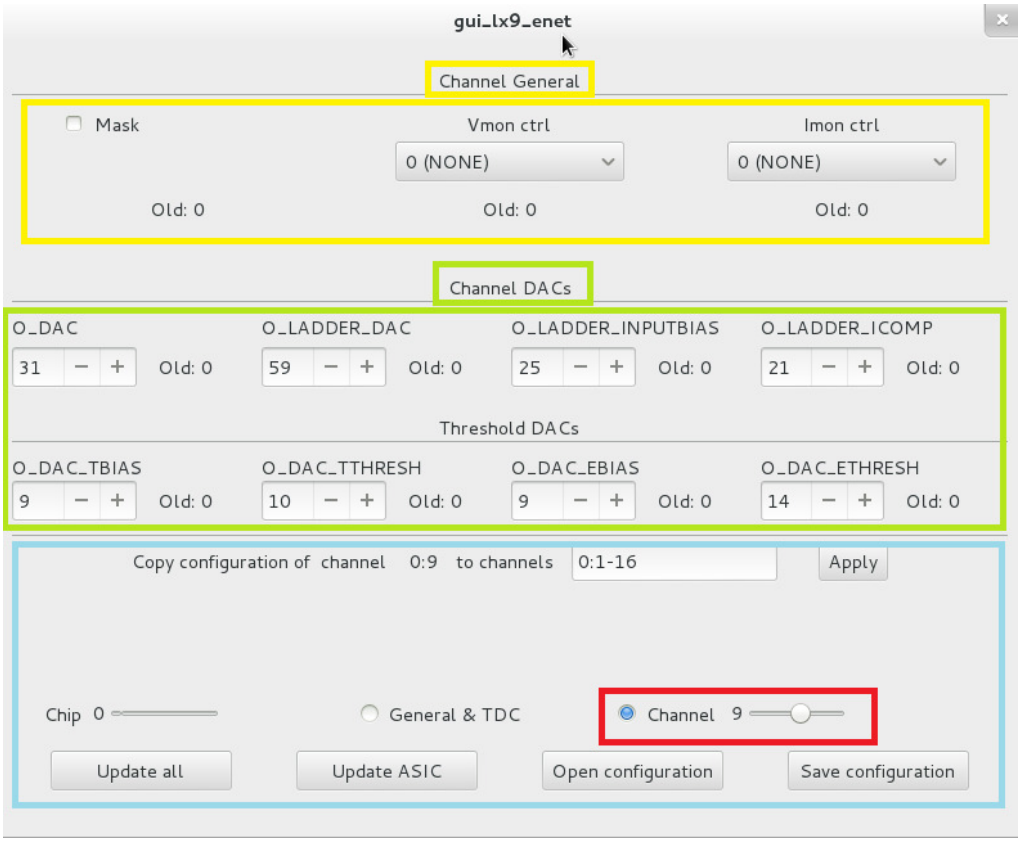


Figure 3.2.: GUI screenshot for "Channel" option.

3.2. DAQ

The data acquisition software is used for operating the STiC. It collects the data and stores it to disk space, and run and configure the online monitoring tool. The data is stored in ROOT file format, named dump.root which contains a TTree with all the data received from the STiC. (If you are not familiar with the TTree class, the full class documentation is in http://root.cern.ch/root/html/TTree.html). The TTree contains the following branches:

- **packet_number** - packet ID the event was transmitted in
- **frame_number** - frame number of the event (for debug)
- **channel** - channel number
- **T_CCM** - time coarse counter master value
- **T_CCS** - time coarse counter slave value
- **T_badhit** - bad hit flag for the time measurement

Table 3.2.: GUI list function for selection of "Channel".

Frame	Name	Value range	Function
	Mask	Select	Selected - channel masked
Channel General	Vmon ctrl	0-8	Activate[1] the monitoring of: 0 - NONE, 1 - T-bias, 2 - T-threshold , 4 - E-bias, 8 - E-threshold
(marked yellow 3.2)	Imon ctrl	0-2	Activate[2] the monitoring of: 0 -NONE, 1 icomp_mon, 2 ibias_mon
	O_DAC	0-63	SiPM bias tuning ($\approx 700\ mV$) in the range 100-800 mV. Vinput must be larger than 200 mV to operate properly.
	O_LADDER_DAC	0-63	Timing and energy scaling DAC.
	O_LADDER_INPUTBIAS	0-63	DAC for the input stage. Reducing this DAC value will increase the tail current and shrinks the energy spectrum.
Channel DACs	O_LADDER_ICOMP	0-63	Setting this DAC value ¿O_LADDER_INPUTBIAS value provide low input Z.
(marked green 3.2)	O_DAC_TTHRESH	0-15	Set timing threshold increases proportionally with this threshold, and also increases the hysteresis.
	O_DAC_TBIAS	0-15	It controls the range of O_DAC_TTHRESH
	O_DAC_EBIAS	0-15	Set energy threshold. The energy threshold increases proportionally with this DAC.
	O_DAC_ETHRESH	0-15	It controls the range of O_DAC_ETHRESH
	Copy of...	0-16	Copy the current configuration to the specified channel number (can be range i-j, or discrete i,j,k..), need to push "apply" to execute (syntax: Chip:Channel number)
	Channel	0-16	only the selected channel is configured
	Chip	0-3	Select the chip you want to configure
Bottom frame	Update all		Update all ASICs with the selected DACs configuration
(marked blue 3.2)	Update ASIC		Update the selected chip only
	Open configuration		Upload configuration file
	Save configuration		Save current configuration to a file

- **T_fine** - fine counter value of the time (improved resolution for the time information)
- **E_CCM** - coarse counter master value for the energy measurement
- **E_CCS** - coarse counter slave value for the energy measurement
- **E_badhit** - bad hit flag for the energy measurement
- **time** - time information of the rising edge
- **energy** - time over threshold value

- **errors** - the flag indicates if the corresponding event stored is good for analysis or not (has errors)

The output root file is used for analysing the data. Most of the branches listed above are used only in the DAQ software. For analysis purpose only the branches named channel, time, energy and errors are used.

When running the DAQ software, the online monitoring tool is being opened automatically. The online monitoring has two windows, Readout Monitor and Monitor GUI.

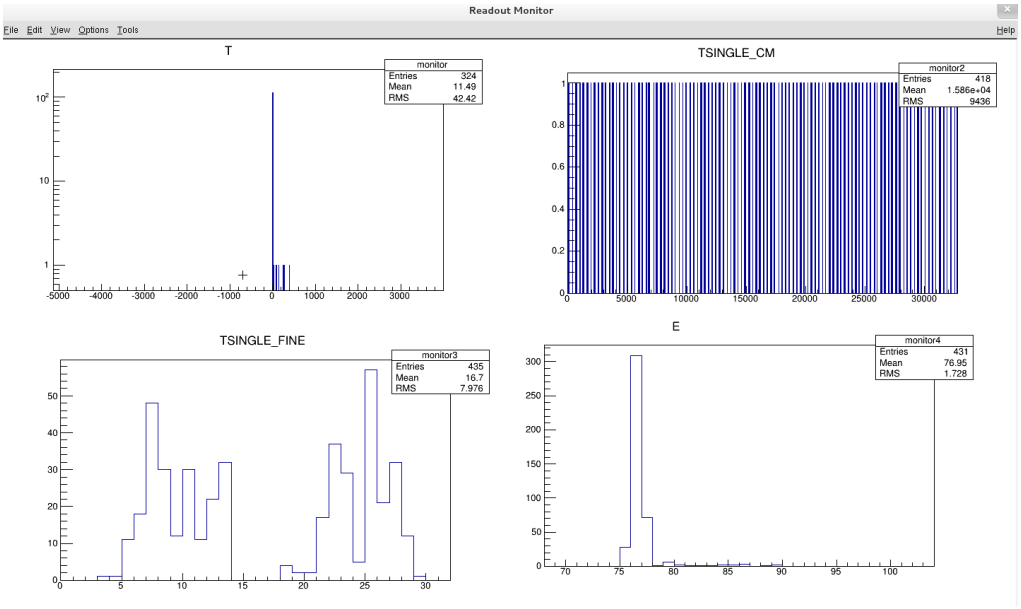


Figure 3.3.: Screen shot of the online monitor histograms.

The Readout Monitor presents 4 histograms which the user can choose. The histograms are being updated with the arrival of new data (only after selecting parameters from the Monitor GUI options, the histogram starts to fill). An example can be seen in Figure 3.3. Each histogram can be scaled during the run using the coarser to shift the axis. The Monitor GUI (can be seen in Figure 3.4) is where the user selects what he wants to display on the Readout monitor from the following options:

- **E** - shows the energy spectra (Time Over Threshold) of the input signal according to the actual thresholds.
- **TSINGLE** - combination of TSINGLE_FINE + TSINGLE_CM (50.2ps bin)
- **TSINGLE_FINE** - fine counter histogram (50.2ps bin)

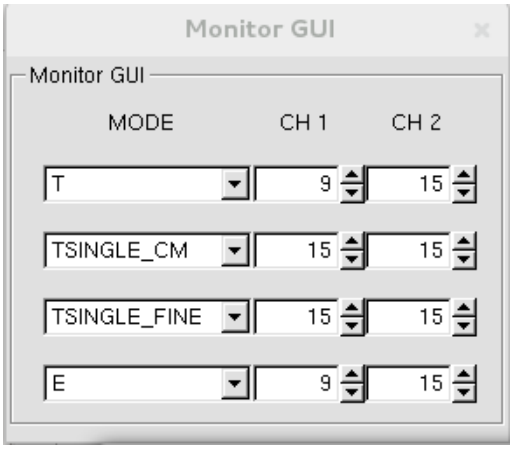


Figure 3.4.: Screen shot of the online monitor GUI.

- **TSINGLE_CM** - coarse-counter histogram (1.6ns bin)
- **CHANNEL** - histogram with the number of hits per channel.
- **BADHIT** - shows the bad hits (the rejected ones)
- **PERIOD** - it allows checking the period of the events in a specified channel. This is especially useful for testing purposes with pulse generator to check that the PLL really locks at the right frequency.
- **T** - time difference between the two specified channels.

In addition there is a reset button at the bottom of the Monitor GUI. This will only reset all displayed histogram, it will not affect the integrated data collected to this point.

Chapter 4.

Step by step

This part of the user guide is a step by setp instruction, from first installation until receiving an output file that is ready for analysis. In addition at the end of this chapter there are hints for doing the analysis.

Before you continue, please check that your PC meets all the requirements that are listed in Section 1.2.

4.1. First installation

4.1.1. Hardware

1. Connect the FPGA to the STiC - J35 (STiC) to J4 (FPGA), and J36 (STiC) to J5 (FPGA).
2. Connect the FPGA to to the HUB, using Ethernet cable (better to use the recommended HUB)
3. Connect the HUB to the PC, using Ethernet cable.
4. Connect the computer and FPGA to the same network by connecting both Ethernet cables from computer and FPGA to a network switch.
5. Connect STiC to the main board - J6, J30, and J31 (STiC) to one of the option connection in the main board , for example J17, J9, and J25.
6. apply power to STiC

(a) If using STiC standalone, use connector J31. The power consumption should be around 250 mA for 1.8 V and 80 mA for 3.2 V. After connecting to the FPGA it will be around 1.8 V (390 mA), 3.2 V (90 mA).

(b) If using the Mainboard for power, apply 12 V via connector J32 on the Mainboard. The current consumption should be about 370 mA when supplied with 11 V.

7. Connect the FPGA power via the USB to the PC (not the microUSB connector).

8. If needed, connect HV cable to J41 (main board), it will give the HV through connector J6 (STiC) to the SiPMs.

The system connected should look like Figure 4.1 (when using the spartan6 lx9 microboard).

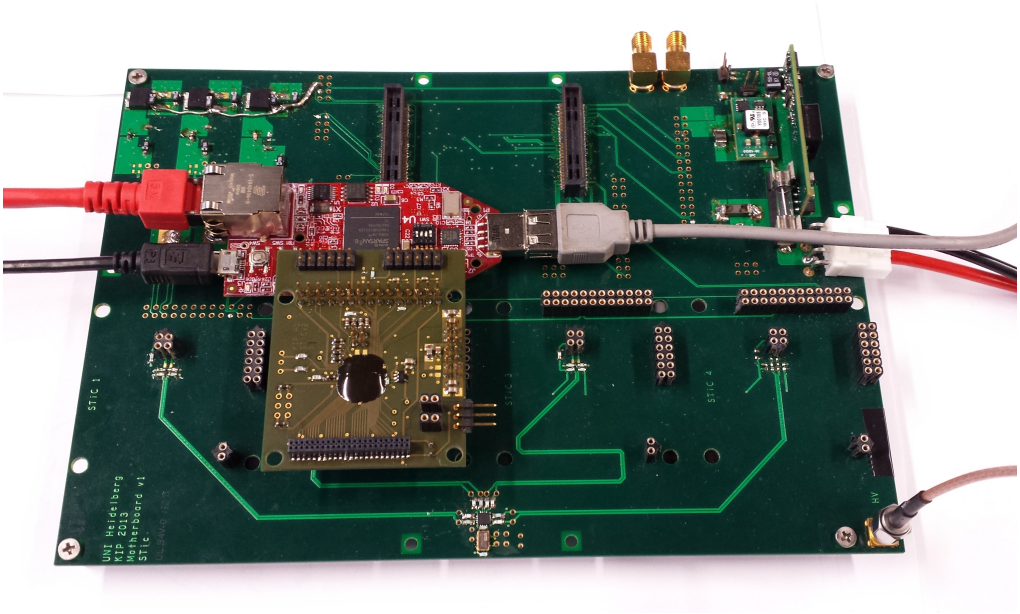


Figure 4.1.: Connection scheme of the FPGA, STiC and the main board.

4.1.2. Software

First you need to setup the workspace. This is the environment where all the folders and files of the package are in. To create a workspace directory in the desired place type:

```
mkdir workspace
cd workspace
```

Copy all the package folders from the CD to the workspace. Now, when typing ls you should see the following folders:

```
ls
chip-interfaces stic2daq stic2gui stic2-mblaze init_stic.sh
```

The script init_stic.sh is used to compile all the different part of the code (in all the folders) you should do it only in the first use or if you change some of the codes. To run the script type:

```
source init_stic.sh
```

It is important to check that there are no errors in the compilation.

4.1.3. FPGA program

Now open a new shell for running the SDK program from Xilinx. This program is used in order to configure the FPGA. Type:

```
cd stic2-mblaze/
source /opt/Xilinx/14.5/ISE_DS/settings64.sh (point to the correct place, it might be installed in different place at your computer)
xsdk (xilinx software installed beforehand)
```

A new window should open for the Xilinx software. A pop up window asks you to select a workspace folder. You should select the folder named SDK_codes (under stic2-mblaze folder) If it is the first use you should close the welcome window.

The software has two main windows (see Figure 4.2) . On the left window, named ”Project Explorer”:

```
right click and select ”import” (see Figure 4.3).
Choose general→Existing projects into workspace
Browse to the folder SDK_codes, and push ”Finish” at the bottom of the window.
```

Now on the left window you will have all the folders that are in the SDK_codes folder. Before we build all projects, it is recommended to clean all the projects.

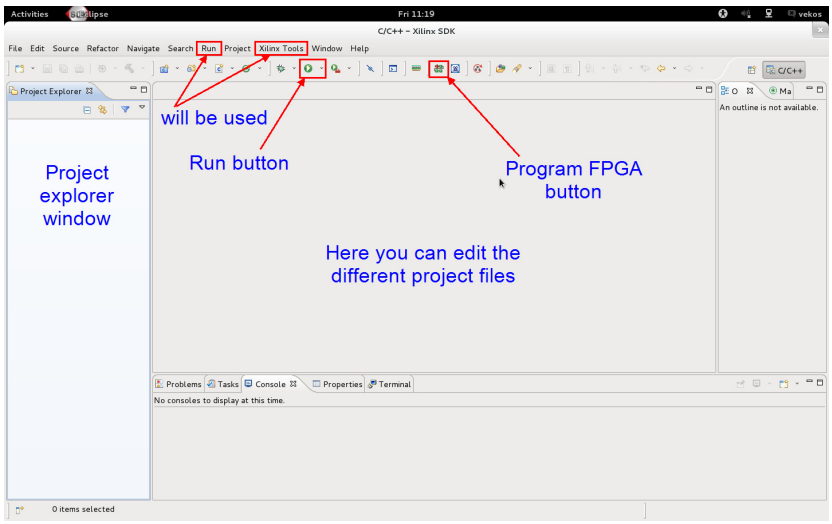
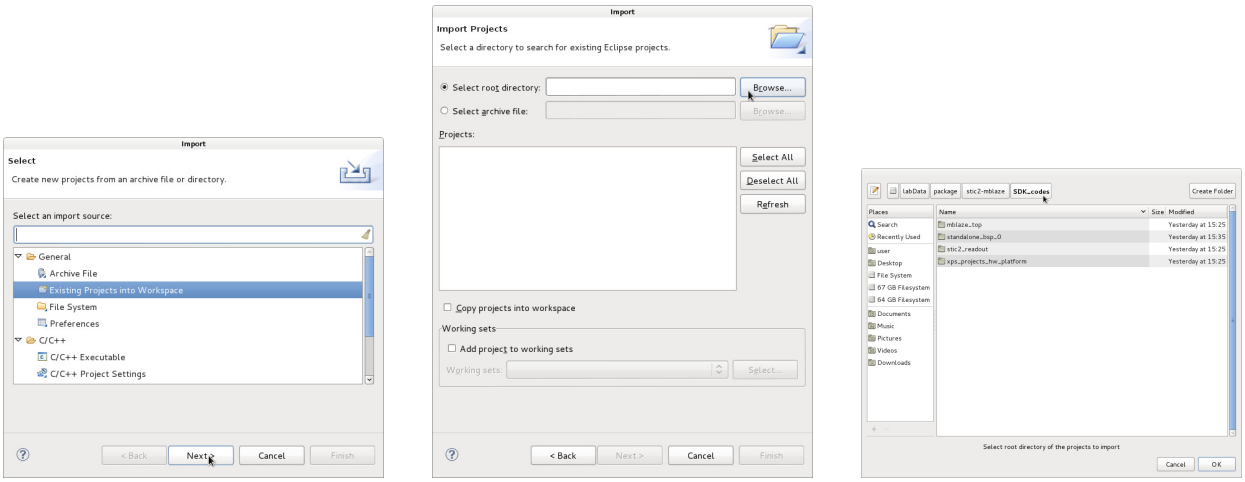


Figure 4.2.: The SDK main window. Buttons and menus that are being used, are marked.



(a) (b) (c)

Figure 4.3.: SDK import menus. (a) general option (b) browse window (c) folder selection menu.

Right click on each of the folder (total of 3) and select clean project (see Figure 4.4).

We need to choose the IP address for the FPGA. In the Project Explorer go to the file main.cc in mblaze_top→src. You can define the IP in lines 111-113 (default value 192.168.0.2).

Now let's build the project:

Right click on standalone_bsp_0→build project.
build the same way mblaze_top.

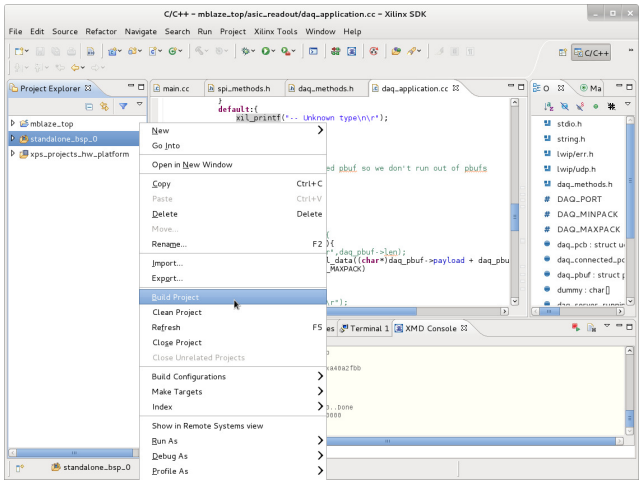


Figure 4.4.: The SDK main window, on the picture are marked the buttons and menus that are been used.

Next step is to set JTAG, it is important to check that you have the correct permission to talk with the USB port. Go to:

Xilinx Tools→Configure JTAG settings (in the top toolbar).

You should see the window as in Figure 4.5. Here type the following selections:

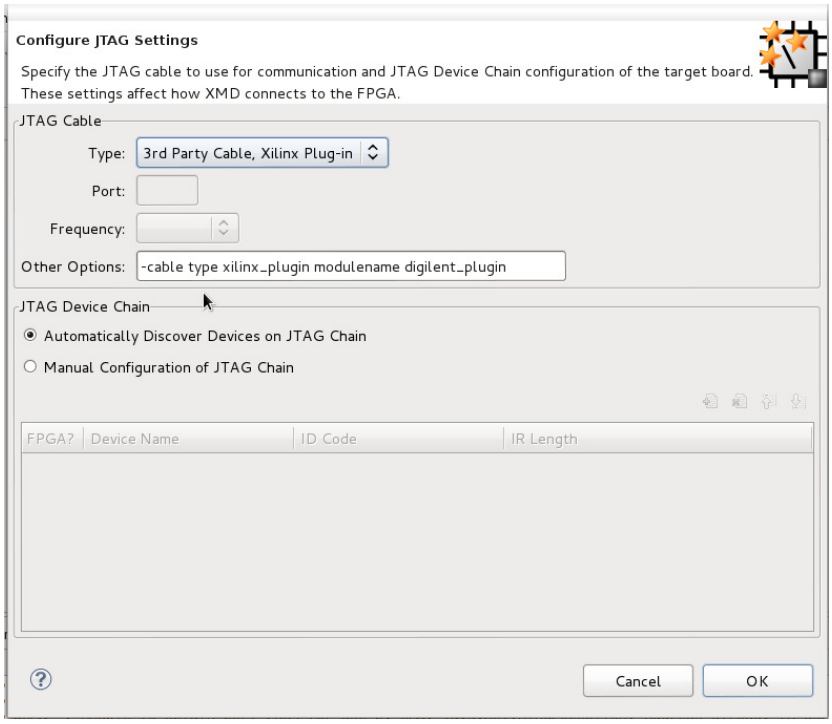


Figure 4.5.: The JTAG configuation menu.

Type: 3rd Party Cable, Xilinx Plug-in
Other Option: -cable type xilinx_plugin modulename digilent_plugin
Select Ok.

Select the button Program FPGA. You should see the window in Figure 4.6 Select the

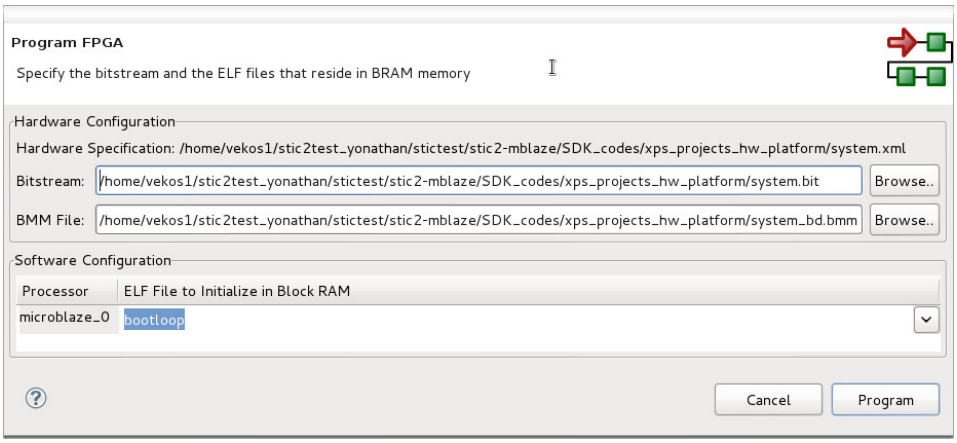


Figure 4.6.: The FPGA program menu.

following:

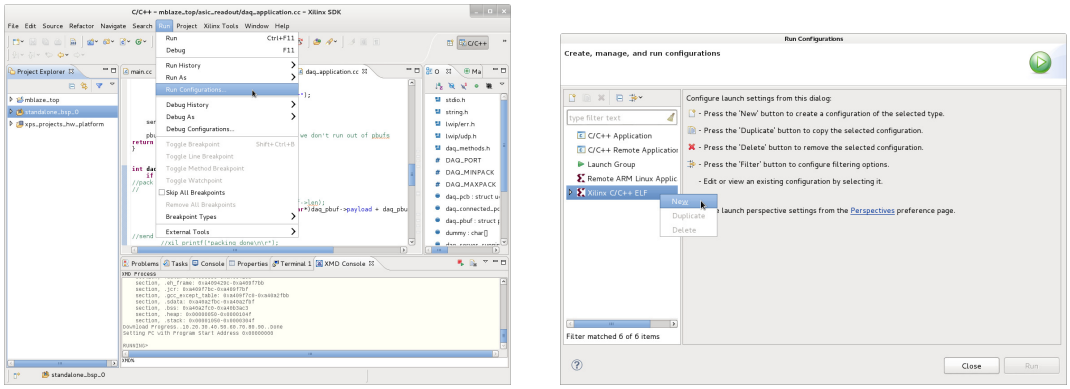
Bitstream:/workspace/stic2-mblaze/SDK_codes/xps_projects_hw_platform/system.bit
BMM File: /workspace/stic2-mblaze/SDK_codes/xps_projects_hw_platform/system_bd.bmm
select bootloop for ELF File to initialize in Block RAM
press PROGRAM

Go to the Run menu and select:

Run configurations... (see Figure 4.7)
On the left window right click on Xilinxx C/C++ ELF
Press New
On the right window you can set the name of the project.
Choose the project mblaze_top and Debug/mblaze_top.elf in the C/C++ Application frame
Push run (this can take a min).

Now you basically saved all the environment, and next time you upload the xsdk in this setup you will have under the Run (green button) your run option with the selected name and you can just run it.

Next step is to setup a new wired connection manually in your PC, different than the one of the FPGA but with the same subnet. For this go the network settings in your PC and define



(a) (b)

Figure 4.7.: SDK run menus. (a) run configuration option (b) create mange and run configurations.

a new connection with the given IP (different instructions depending on the system you are running).

Please check that the connection is ok by typing in the shell

```
ping 192.168.0.2 (the IP of the FPGA),
```

If data is streaming , it is ok, continue to the next stage.

4.2. Running the DAQ

Now we can run the DAQ software, for this purpose go back the previous shell (different than the one running SDK).

```
cd stic2daq
./dump
```

This will start the DAQ software. When ending the daq, all data collected will be saved to ROOT file named ”dump.root” (see details in Section 3.2).

4.3. Using GUI

To run the GUI just type:

```
cd ../stic2gui
source start_gui.sh
```

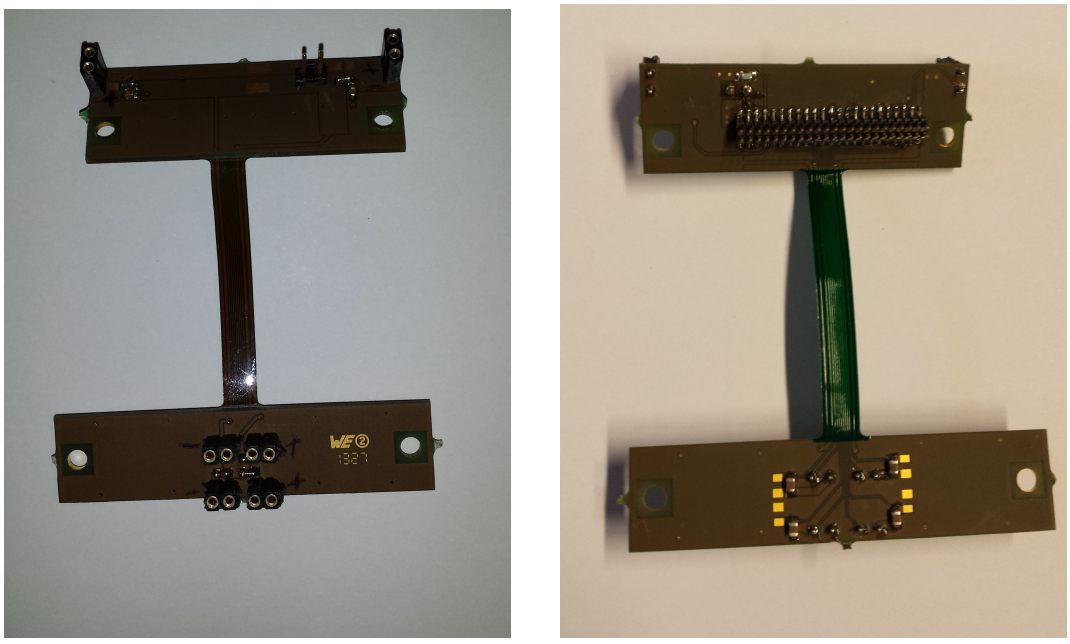
The window of the GUI should appear.
Important: please check inside the file start_gui.sh that the IP address is the same as you selected (by default 192.168.0.2).
Now the GUI is running and you can test if everything is connected correctly.

Go to Section 3.1 for all the details about the different functions of the GUI.

4.4. Pulse Injection test

This part is not mandatory, but is highly recommended. Before starting doing measurements with your detector, it is better to check that the system is working properly with a known signal.

For this purpose additional connection is required to connect the pulse positive signal to the STiC input signal (connector J32 in the STiC test board). The flex print connector for this use is shown in Figure 4.8. When using a pulse generator, the signal should pass it through a



(a) (b)

Figure 4.8.: Flex connector for the testing. (a) upper side (b) bottom side.

capacitor (100 pF recommended) to the positive input of a channel, and connect it also to one of the GNDs (connector J31 in STiC test board), see Figure 4.9 for the exact setup.

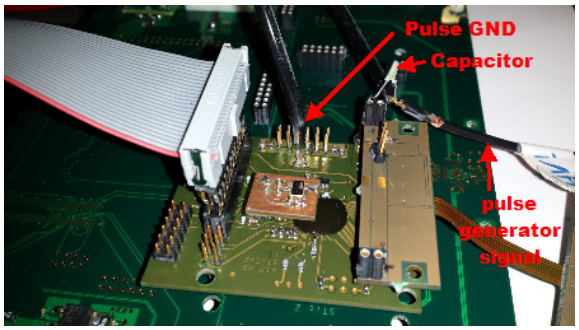


Figure 4.9.: The setup for pulse injection measurement.

For testing you could upload setup file from the GUI:

”Open configuration”
choose the file named ”STiC2Config_test_charge.txt”.

For this configuration we used a pulse with the following setup:

- period 599.47 ns
- amplitude 408 mV
- pulse width 200 ns
- edge time 5 ns
- 20 cycles
- burst period 10 ms
- capacitor of 120 pF

For this setup you should see the distributions plotted in Figure 4.10. You can verify that the period is the same as generated by the pulse generator (remember that each TDC count is 50.2 ns).

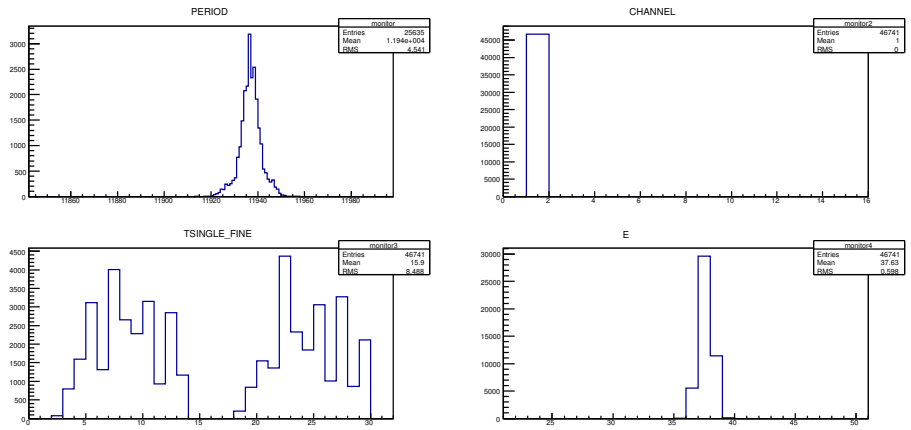


Figure 4.10.: Online monitoring view for the pulse injection test.

After closing the online monitor of the DAQ, a root file is saved (it's recommend to close the online monitor via File→ Quit Root, to void error during closing the DAQ). In this file you can find also the different plots of the online monitoring, which you can preformed on them a fast analysis in order to check that the system is working properly. You can open the file using ROOT, whith the following commands:

root
TBrowser y
Double click on the file name - Figure 4.11(a)
Double click on the histogram of the period - Figure 4.11(b)
Select the Fit Panel (under Tool menu) - Figure 4.11(c)
Fit to a gaus (default setup) and press Fit - Figure 4.11(d)

The fit result for the period is shown in Figure 4.12. The period must be confirmed with the input injected signal. In this example the mean value from the fit is 1.194×10^4. So the period is:
$1.194 \times 10^4 \times 50.2 \times 10^{-12}$ (tdc count is 50.2 ps) = 599.4 ns.

Important: if the your period is not correct, that is probably because the PLL is not locked. The procedure for locking the PLL is the following:

In the GUI DACs frame (see Figure 3.1) change the value of VPCP/VCascP and VNVCODelay to 63
Press Update ASIC (this will unlock PLL)
Change the values of VPCP/VCascP and VNVCODelay back to zero
Press Update ASIC (this will lock PLL)

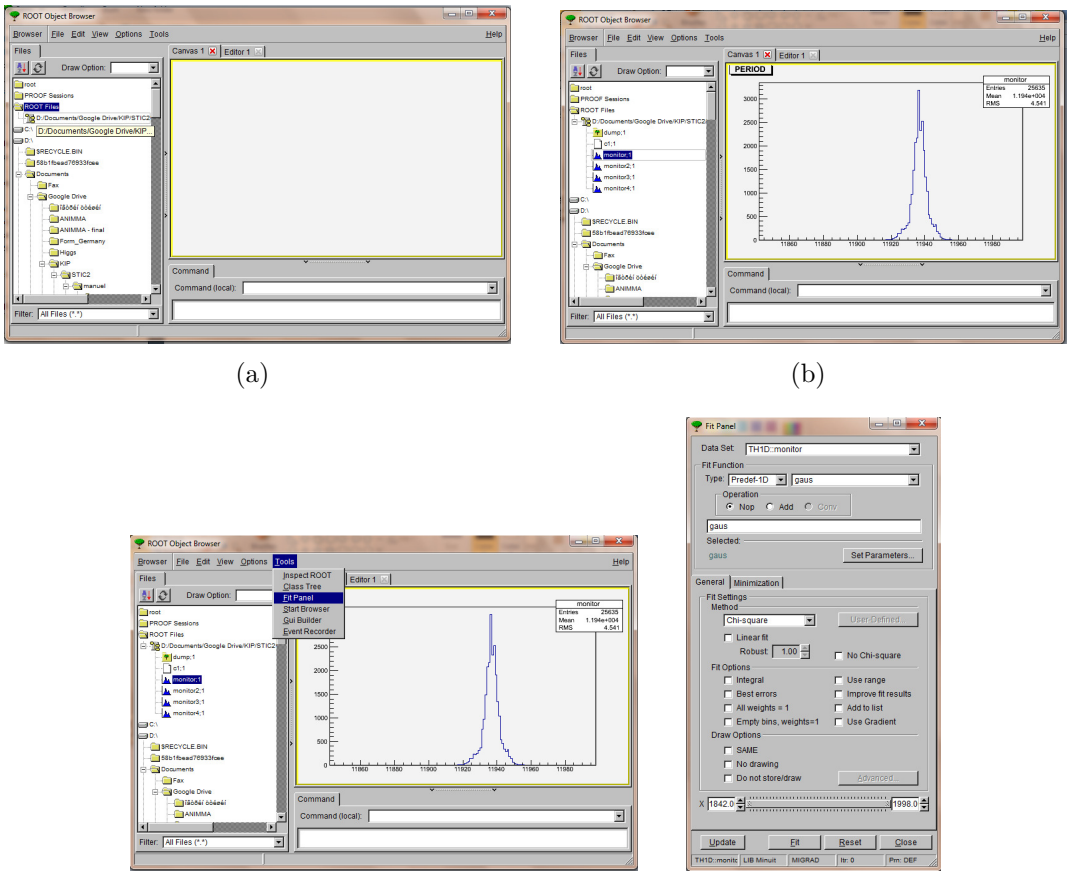


Figure 4.11.: Root fast analysis. (a) step 1 (b) step 2 (c) step 3 (d) step 4 .

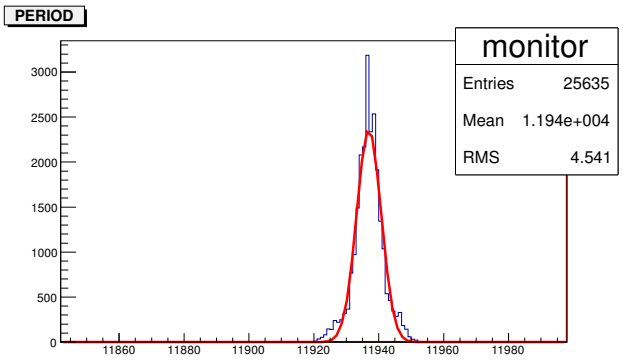


Figure 4.12.: Fit to the period histogram.

Now repeat the measurement describe above and check the period is the correct one.

Now when you are sure that everything is working correctly you can start your measurements with your detector.

4.5. CTR measurement

For CTR measurement we will use the same setup as in the previous section, with a few modifications. Instead of the pulse generator signal, we will connect directly two MPPCs to two channels in the flex connector. When using a detector, the capacitor is not needed any more.

The setup now will look like in Figure 4.13

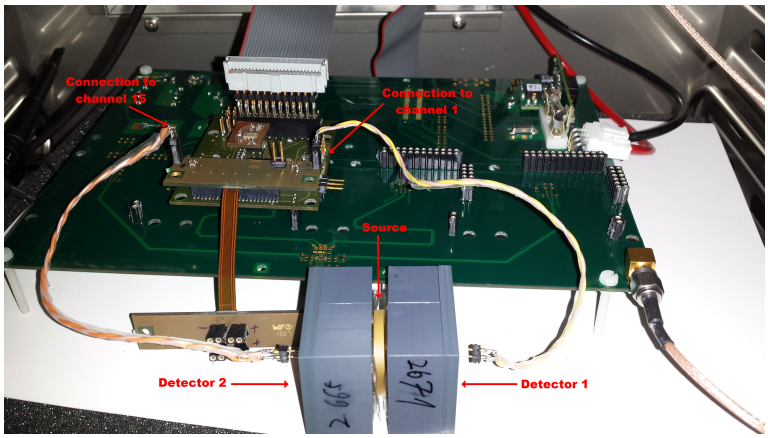


Figure 4.13.: Setup for CTR measurement.

On the the MPPC cover the recommended V_{op} is written. In our case, when using the STiC chip it is recommended to set the HV around 1 V higher. For finding the optimum HV value, you should repeat the following measurement for different HV values (HV scan).

Now you should set the different DACs values until you see a good energy spectrum in the histogram, and that all other distributions are ok.

After you collect all the required data, you could do the CTR measurement. In our example we searched for coincidence between two detectors and we fill the histogram with the values of t1-t2 (arrival time difference between the two detectors). Because we aim to measure the coincidence of the 511 keV electron, we will use the energy spectrum in order to select only events that are around this region.

In Figure 4.14 the two detectors energy spectrum are plotted, with the corresponding energy selection. Also plotted is the results for the CTR measurement for testing two $3 \times 3mm^2$ MPPCs with HV of 73.5 V.

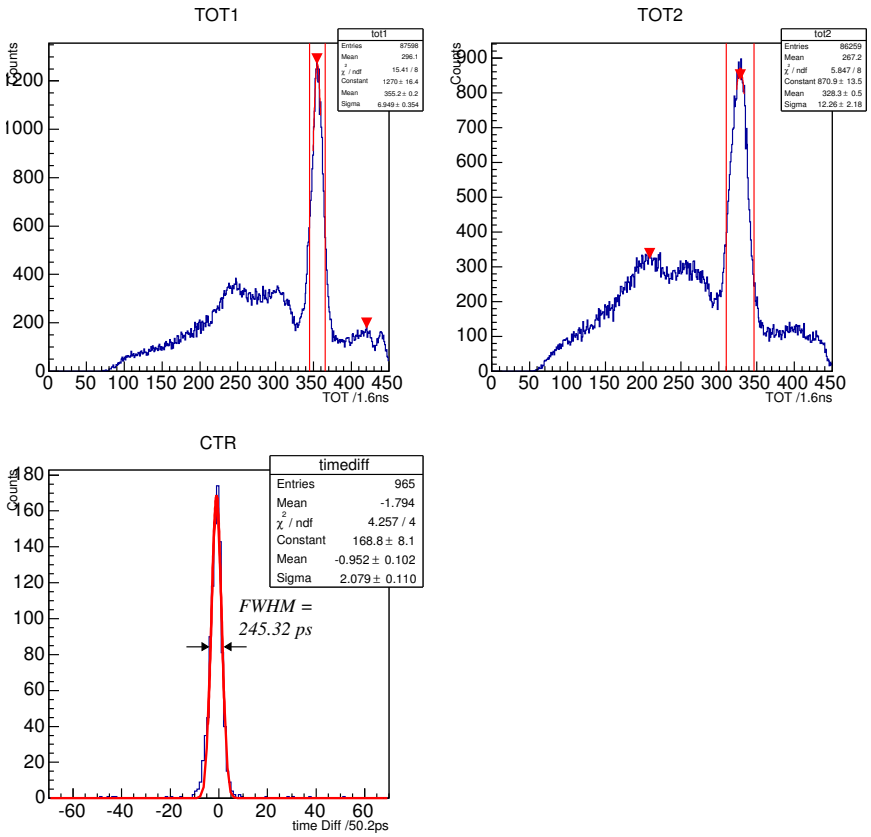


Figure 4.14.: CTR measurement results. Top two plots are the energy spectrum for the two detectors. The red lines correspond to $\pm 1.5\sigma$ around the 511 keV peak. The bottom plot is the CTR distribution for the related energy.

Appendix A.

Test board connectors

Table A.1.: Details of J32 input connector.

Input	Pin	Pin	Input
Analog GND	1	24	Analog GND
Channel 15 (negative)	2	25	Channel 15 (positive)
Channel 13 (negative)	3	26	Channel 13 (positive)
Channel 11 (negative)	4	27	Channel 11 (positive)
Channel 9 (negative)	5	28	Channel 9 (positive)
Analog GND	6	29	Analog GND
Channel 14 (negative)	7	30	Channel 14 (positive)
Channel 12 (negative)	8	31	Channel 12 (positive)
Channel 10 (negative)	9	32	Channel 10 (positive)
Channel 8 (negative)	10	33	Channel 8 (positive)
Analog GND	11	34	Analog GND
Channel 6 (negative)	12	35	Channel 6 (positive)
Channel 4 (negative)	13	36	Channel 4 (positive)
Channel 2 (negative)	14	37	Channel 2 (positive)
Channel 0 (negative)	15	38	Channel 0 (positive)
Analog GND	16	39	Analog GND
Channel 7 (negative)	17	40	Channel 7 (positive)
Channel 5 (negative)	18	41	Channel 5 (positive)
Channel 3 (negative)	19	42	Channel 3 (positive)
Channel 1 (negative)	20	43	Channel 1 (positive)
Analog GND	21	44	Analog GND
Digital GND	22	45	Digital GND
HV GND	23	46	HV GND

Table A.2.: Details of J31 input power connector.

Input	Pin
V_{CC} 1.8 V analog	5,7
V_{CC} 1.8 V digital	1,3
V_{CC} 3.3 V digital	9,11
Digital GND	2,4,10,12
Analog GND	6,8

Table A.3.: Details of J6 interface connectors.

Input	Pin	Pin	input
Digital GND	24	23	Digital GND
Do_FIFO_EMPTY (debug only)	22	21	Dummy
DO_ANY_DREADY (debug only)	20	19	DO_FIFO_FULL (debug only)
Digital GND	18	17	Digital GND
Serial clock (SPI)	16	15	Digital reset
Digital GND	14	13	Serial data output (SPI)
Digital GND	12	11	Digital GND
Readout clock input (positive) (recommended 160 MHz)	10	9	Readout clock input (negative) (recommended 160 MHz)
Digital GND	8	7	Digital GND
Serial data input (SPI)	6	5	Chip select (Cs)
data transmission (Txd_N)	4	3	data transmission (Txd_P)
not used	2	1	Digital GND

Table A.4.: Details of J25 input TDC test signal.

Input	Pin
Time positive	1
Time negative	2
Energy positive	5
Energy negative	6
Analog GND	3,4

Table A.5.: Details of J19 input analog test signal.

Input	Pin
Input positive signal	1
Analog GND	2
Input negative signal	3

Table A.6.: Details of J24 output analog test signal.

Input	Pin
Output time positive signal	1
Output time negative signal	2
Analog GND	3,4
Output energy positive signal	5
Output energy negative signal	6

Table A.7.: Details of J30 input TDC clk signal.

Input	Pin
Input PLL negative	1
Digital V_{cc} 3.3 V	2
Input PLL positive	3
Digital GND	4

Appendix B.

Technical specification

Table B.1.: STiC2 technical specification.

Parameter	Value	Units	Condition
Charge injection jitter analog channel	< 30	ps σ	@3 pC input charge, C=33 pF, using Tektronix AWG7102
Charge injection jitter full chain	43	ps σ	@ 6.6 pC input charge, C=33pF, using Tektronix AWG7102
DAC range	0.7	V	
Charge response	Linear	-	For input charge larger than 3 pC
Power consumption	19	mW/ch	@ room temperature
SPTR[1] (single ended mode)	180	ps σ	Using a fast laser system (PILAS PiL063SM) and Hamamatsu MPPC S10362-11-100 (1x1mm^2, 100μm x 100μm pixel)
SPTR (differential mode)	180	ps σ	Using a fast laser system (PILAS PiL063SM) and Hamamatsu MPPC S10362-11-100 (1x1mm^2, 100μm x 100μm pixel)
Energy resolution	11	%	
CTR[2] (differential)	220	ps FWHM	@T = 18°C, using crystal LSO 3.1x3.1x15mm^3, ^{22}Na source, Hamamatsu SiPM ref: S10362-33-050C. Bias operation voltage=74 V

[1]Single Pixel Time Resolution
[2]Coincidence Time Resolution